trainee: Florent N'Kada-Menyé
supervisor: Federico Stagni

Study and improve DIRAC's DISET framework

May 29, 2017

What is it about ?

- Studying DIRAC by assessing the performances of a standard DISET service
  - We create a service that returns a small dictionary, in which every value equals "fibonacci(1)".
  - We create a client, that spawns a certain amount of threads, all trying to connect to the service at the same time.
  - We measure the amount of time taken by the DISET service to process all the requests.

Comparing DISET with Python 2.7's xmlrpc library

- We used 3 types of rpc servers:
  - A thread server: incoming rpc requests are assigned to threads in a thread pool
  - A process server: uses forking
  - A basic server: processes the requests one after the other
- They are compared to a standard DIRAC service
  - using a single thread (option MaxThreads=1)
  - using multithreading (option MaxThreads=15)

Comparing DISET with Python 2.7's xmlrpc library

- Each server has a method that returns a small dictionary of 100 values, all equal fibonacci(1).
- We first run a client script spawning 1 thread, then 75 threads, then 150... until 1500 threads.
- Each client script connects to the server, then requests a dictionary.
- Why 1500 threads max ? Beyond 1500 client threads, the basic server has issues processing them all and sometimes, some client threads get disconnected from the server before their request has been processed.

First results

The 3 types of xmlrpc servers

Figure 1: Amount of time taken, in seconds, to process all the clients' requests depending on the number of client threads

The 3 types of xmlrpc servers

- The basic server takes so much time than we cannot properly compare the other servers.
- For the basic server, the amount of time taken does not increase regularly. Given a certain number of clients, it can be very fast of take a lot of time.
- In general, the amount of time taken increases with the number of clients.
- The basic server can take until 6 minutes when there are more than 1000 clients.

The thread server and the process server

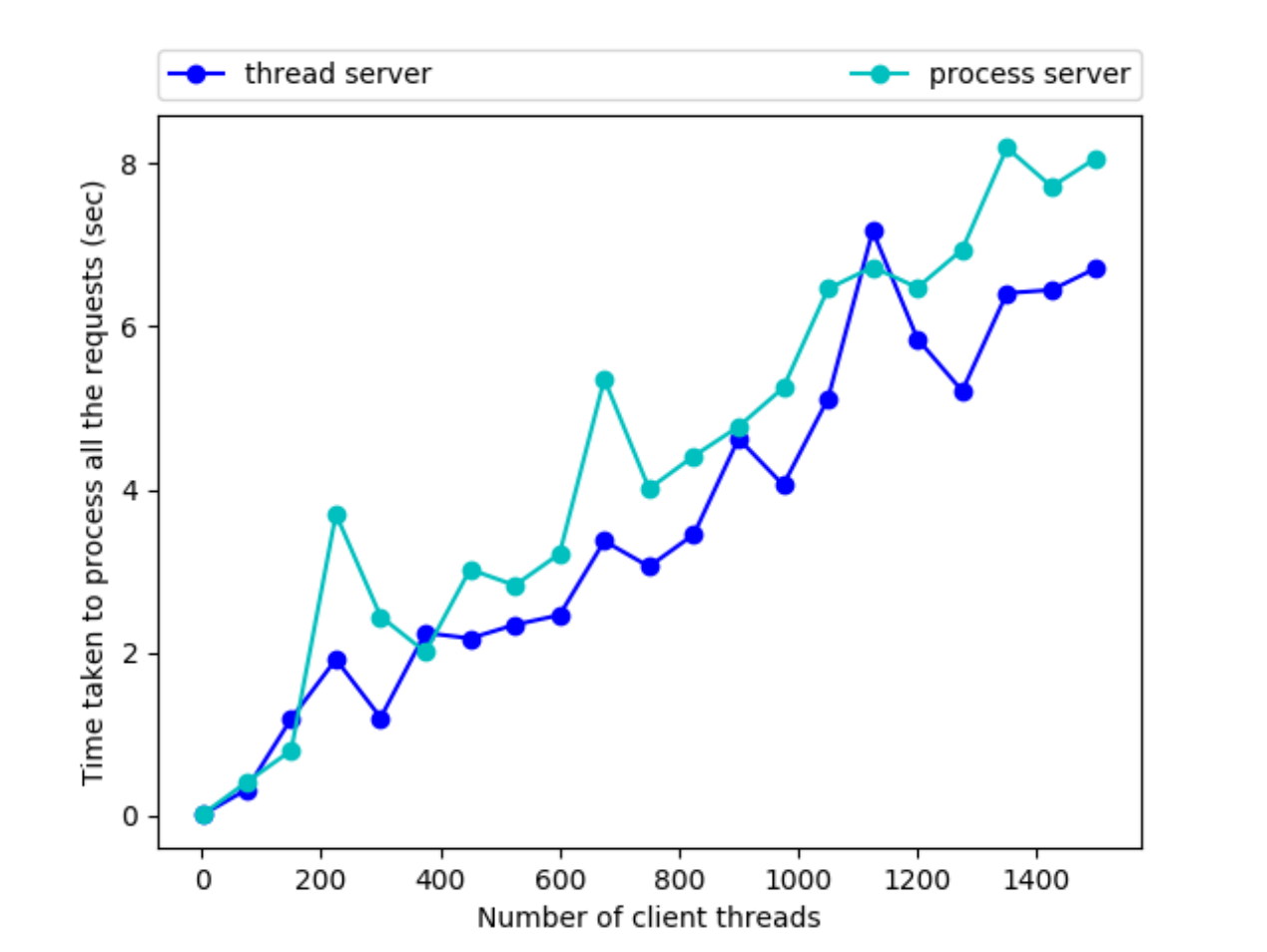

Figure 2: Amount of time taken by the thread server and the process server depending on the number of client threads

The thread server and the process server

- Both of these servers are much faster than the basic one.
- In our case, the thread server seems to be a little faster than the process server. This is because the dictionary returned is relatively small. When the data volume is much bigger, the process server is faster than the thread server.

xmlrpc compared to DISET

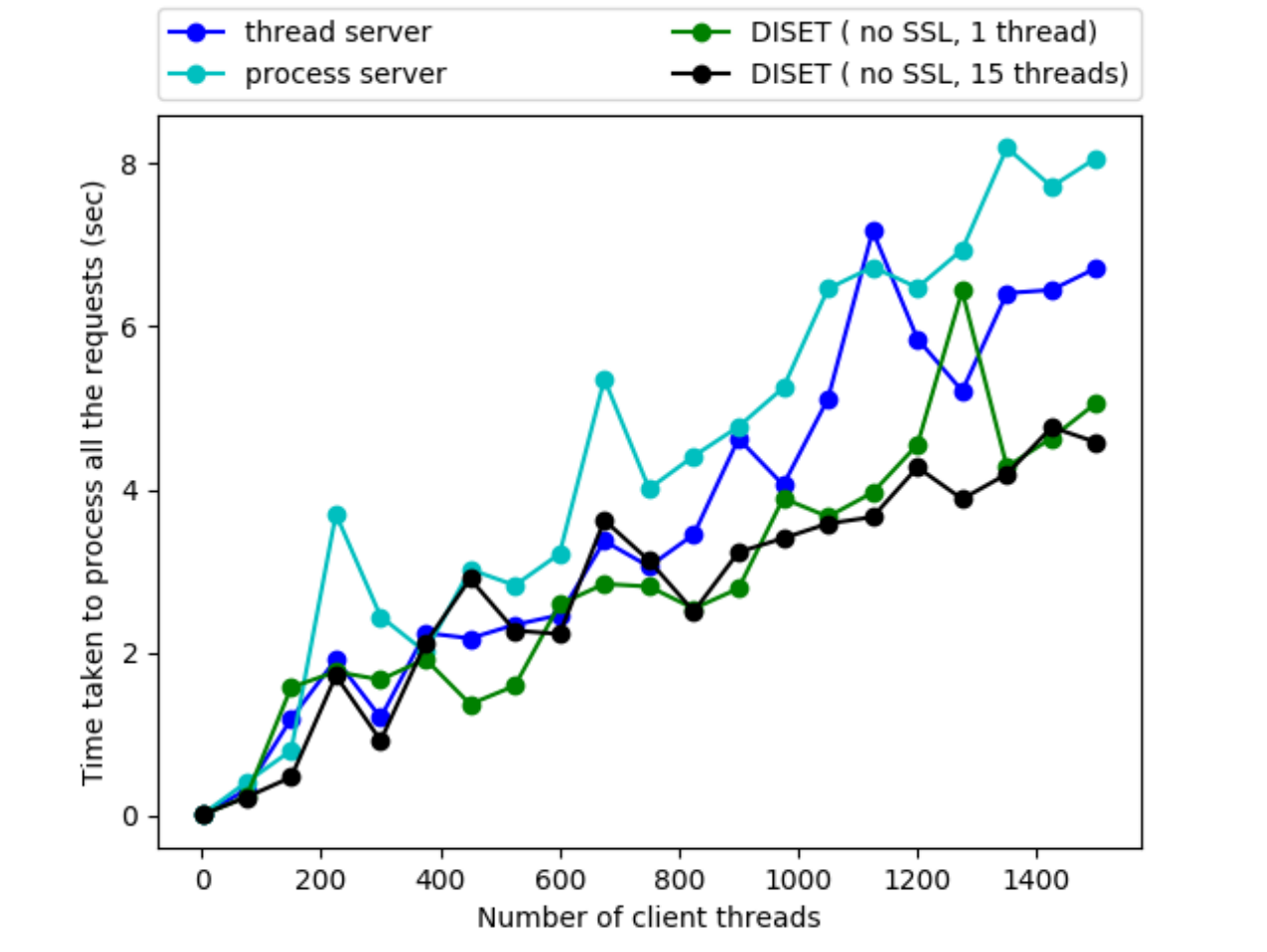

Figure 3: The thread and process servers compared to a standard DISET service

xmlrpc compared to DISET

The DISET service seems to be a little faster than the other xmlrpc servers.

Given the relatively small volume of data processed by the server, there is not much difference between single threading and multithreading in the DISET service.

What's next ?

These are just the first results.

Next steps:

- Run the same comparison, but with fibonacci(30) instead of fibonacci(1).
- Compare the servers, but processing a huge "dictionary of dictionaries of dictionaries".
- When the amount of data is much higher, is DISET really more efficient than the xmlrpc library ?
- Does multithreading bring more efficiency to the server ? Doesn't it take too much time to spawn and close too many threads ?

Thank you for listening